Implementation of Binary Counters using Reversible Logic Gates

C. Santhi[1], Dr. Moparthy Gurunadha Babu[2]
[1]*PhD Scholar, Shri Jagdishprasad Jhabarmal Tibrewala University, Rajasthan, INDIA*
[2]*Professor, Department of ECE, CMR Institute of Technology, Hyderabad, T.S, INDIA*

Abstract- High speed, efficient addition of multiple operands is an essential operation in any computational unit. The speed and power efficiency of multiplier circuits is of critical importance in the overall performance of microprocessors. Multiplier circuits are an essential part of an arithmetic logic unit, or a digital signal processor system for performing filtering and convolution. In this paper, a 7:3 counter circuit that accepts 7 bits of equal weight and counts the number of “1” bits is implemented using reversible logic gates. The7:3 and 6:3 counter circuits can be constructed using full and half adders. In VLSI simulations, the proposed counters are 30%faster than existing parallel counters and also have the merit of equal fanin and fanout using reversible logic gates.

I. INTRODUCTION

Reversible computing is a model of computing where the computational process to some extent is reversible, i.e., time-invertible. In a model of computation that uses deterministic transitions from one state of the abstract machine to another, a necessary condition for reversibility is that the relation of the mapping from (nonzero-probability) states to their successors must be one-to-one. Reversible computing is a form of unconventional computing. A process is said to be physically reversible if it results in no increase in physical entropy; it is isentropic. There is a style of circuit design ideally exhibiting this property that is referred to as charge recovery logic, adiabatic circuits, or adiabatic computing. Although in practice no non-stationary physical process can be exactly physically reversible or isentropic, there is no known limit to the closeness with which we can approach perfect reversibility, in systems that are sufficiently well isolated from interactions with unknown external environments, when the laws of physics describing the system's evolution are precisely known. Probably the largest motivation for the study of technologies aimed at actually implementing reversible computing is that they offer what is predicted to be the only potential way to improve the computational energy efficiency of computers beyond the fundamental von Neumann-Landauer limit [1] of kT ln (2) energy dissipated per irreversible bit operation. Although the Landauer limit was millions of times below the energy consumption of computers in the 2000s and thousands of times less in the 2010s, proponents of reversible computing argue that this can be attributed largely to architectural overheads which effectively magnify the impact of Landauer's limit in practical circuit designs, so that it may prove difficult for practical technology to progress very far beyond current levels of energy efficiency if reversible computing principles are not used. As was first argued by Rolf Landauer of IBM, in order for a computational process to be physically reversible, it must also be logically reversible. Landauer's principle is the rigorously valid observation that the oblivious erasure of n bits of known information must always incur a cost of nkT ln (2) in thermodynamic entropy. A discrete, deterministic computational process is said to be logically reversible if the transition function that maps old computational states to new ones is a one-to-one function; i.e. the output logical states uniquely determine the input logical states of the computational operation. Landauer's principle (and indeed, the second law of thermodynamics itself) can also be understood to be a direct logical consequence of the underlying reversibility of physics, as is reflected in the general Hamiltonian formulation of mechanics and in the unitary time-evolution operator of quantum mechanics more specifically. The implementation of reversible computing thus amounts to learning how to characterize and control the physical dynamics of mechanisms to carry out desired computational operations so precisely that we can accumulate a negligible total amount of uncertainty regarding the complete physical state of the mechanism, per each logic operation that is performed. In other words, we would need to precisely track the state of the active energy that is involved in carrying out computational operations within the machine, and design the machine in such a way that the majority of this energy is recovered in an organized form that can be reused for subsequent operations, rather than being permitted to dissipate into the form of heat. For computational processes that are nondeterministic (in the sense of being probabilistic or random), the relation between old and new states is not a single-valued function, and the requirement needed to obtain physical reversibility becomes a slightly weaker condition, namely that the size of a given ensemble of possible initial computational states does not decrease, on average, as the computation proceeds forwards.

II. REVERSIBLE GATES

Feynman gate is a 2×2 one through reversible gate as shown in figure 1. The input vector is I (A, B) and the output vector is O (P, Q). The outputs are defined by P=A, Q=A⊕B. Quantum cost of a Feynman gate is 1. Feynman Gate (FG) can be used as a copying gate. Since a fan-out is not allowed in reversible logic, this gate is useful for duplication of the required outputs.

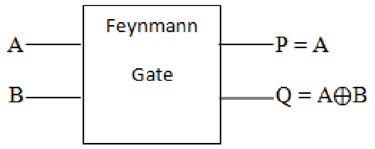


Figure 1: Feynman Gate

A	B	C	B
0	0	0	0
0	1	0	1
1	0	1	1
1	1	1	0

Table 1: Truth table of Feynman gates

Double Feynman Gate (F2G)

Fig.2 shows a 3×3 Double Feynman gate. The input vector is I (A, B, C) and the output vector is O (P, Q, and R). The outputs are defined by P = A, Q=A⊕B, R=A⊕C. Quantum cost of double Feynman gate is 2.

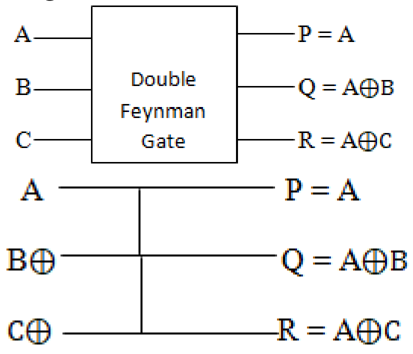


Figure 2: Double Feynman gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	1
1	0	1	1	1	1
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	0	0

Table 2: Truth table of double Feynman gates

Toffoli Gate:

Fig. 3 shows a 3×3 Toffoli gate. The input vector is I (A, B, C) and the output vector is O (P, Q, and R). The outputs are defined by P=A, Q=B, R=AB⊕C. Quantum cost of a Toffoli gate is 5.

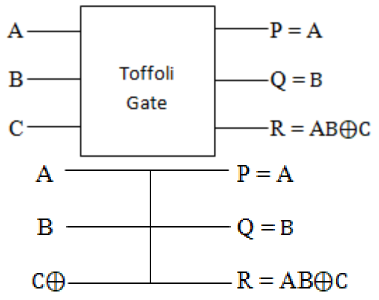


Figure 3: Toffoli gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	0	0
1	1	0	1	1	1
1	1	1	1	1	0

Table 3: Truth table of Toffoli gate

Fredkin Gate

Fig. 4 shows a 3×3 Fredkin gate. The input vector is I (A, B, C) and the output vector is O (P, Q, and R). The output is defined by P=A, Q=A′B⊕AC and R=A′C⊕AB. Quantum cost of a Fredkin gate is 5.

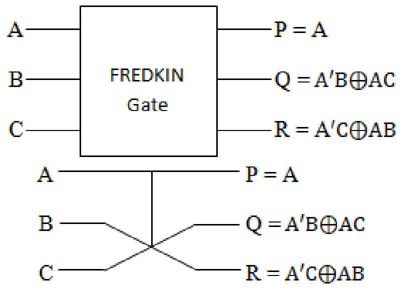


Figure 4: Fredkin Gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	0	0
1	0	1	1	1	0
1	1	0	1	0	1
1	1	1	1	1	1

Table 4: Truth table of fredkin gate

Peres Gate

Fig 5 shows a 3×3 Peres gate. The input vector is I (A, B, C) and the output vector is O (P, Q, and R). The output is defined by P

= A, Q = A⊕B and R=AB⊕C. Quantum cost of a Peres gate is 4. In the proposed design Peres gate is used because of its lowest quantum cost.

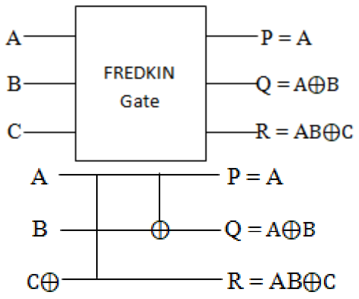


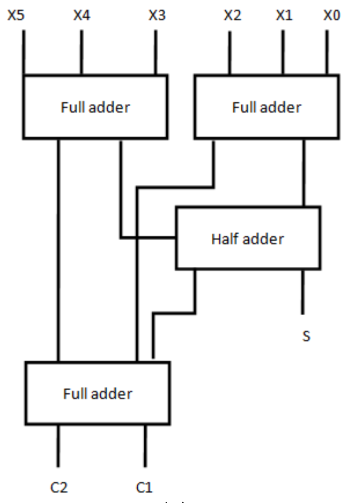
Figure 5: Peres Gate

A	B	C	P	Q	R
0	0	0	0	0	0
0	0	1	0	0	1
0	1	0	0	1	0
0	1	1	0	1	1
1	0	0	1	1	0
1	0	1	1	1	1
1	1	0	1	0	1
1	1	1	1	0	0

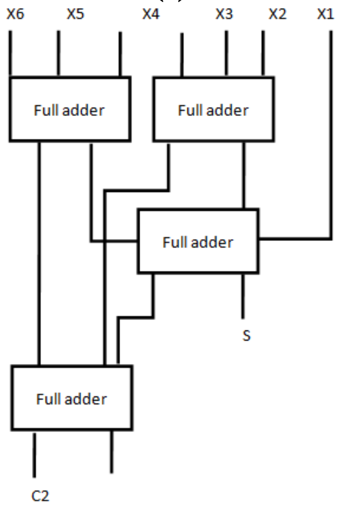
Table 5: Truth table of Peres gate

III. COUNTER CIRCUIT

For any computational unit the essential operation would be of High speed and an efficient method of adding multiple operands. The overall performance for a microprocessor depends on the speed and power efficiency of multiplier circuits [2][3]. The essential part for a multiplier circuit is an arithmetic logic unit or it could be a digital signal processor system to perform filtering and convolution. To get the final product in binary multiplication of integers or fixed point numbers the partial products must be added. To dominate the latency and power consumption the partial products must be added. The efficient way to combine the partial products, compression of columns is used generally. Lot of methods is there to modify the realization of the partial product summation [4][5]. All these methods use full adders which function as counters. By using carry-save adder tree the counters reduce groups of 3 bits having same weight to 2 bits having different weights in parallel. After passing through many layers of reduction process, the total summands are reduced to two, which are later added using a conventional adder circuit. To obtain higher efficiency, large number of bits having equal weights can be considered. The underlying methodology while handling large numbers of bits is the same, that is, bits of one column are counted, producing very few bits of different weights [6]. For example, a 7:3 counter circuits accepts 7 bits having equal weight and computes the number of "1" bits. This count goes to the output which uses 3 bits of increasing weight. Fig 1 shows the construction of 7:3 and 6:3 counter which uses full adder and half adder.



(a)



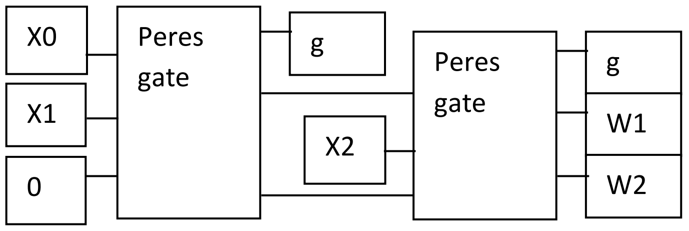
(b)

Fig 6 (a-b): 7:3 counter and 6:3 counter constructed from full and half adders

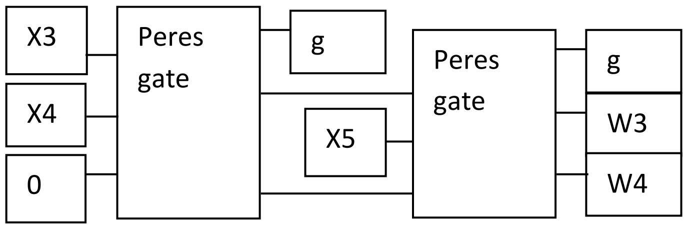
In brief, a counting method has been presented which uses bit stacking circuit followed by an innovative method that combines two small stacks to form a larger stack [7]. By using this method a 6:3 counter has been developed, which does not use XOR gates or multiplexers in its critical path [8]. The simulation results obtained by using our 6:3 counter shows that it is at least 30% faster than the existing counter designs, this method also uses less power. Different simulation results were obtained after running the counter on multiplier designs having various sizes. The usage of the proposed counter circuit has improved the multiplier efficiency for larger circuits, generating 64 bit and 128 bit multipliers which are fast and which consumes less power than other counter based Wallence (CBW) designs.

IV. EXPERIMENTAL RESULTS

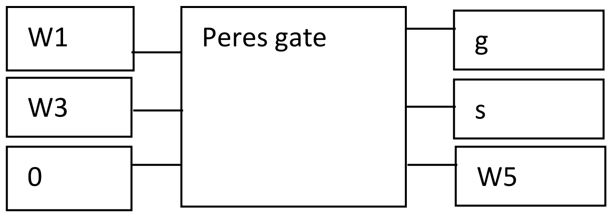
The Verilog implementation of the proposed circuits is implemented as per the circuits presented in the figures 7 and 8.



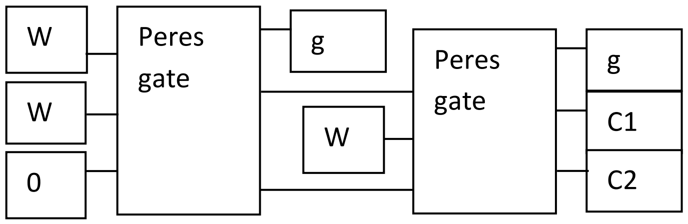
(a) Full adder F1



(b) Full adder F2



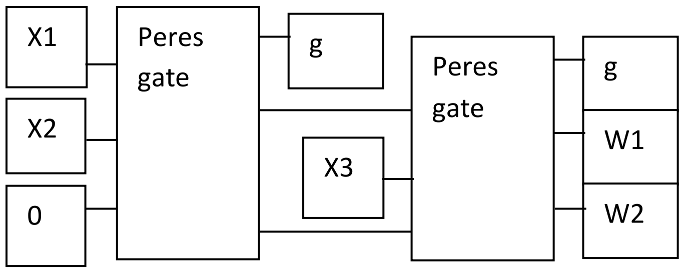
(c) Half adder H1



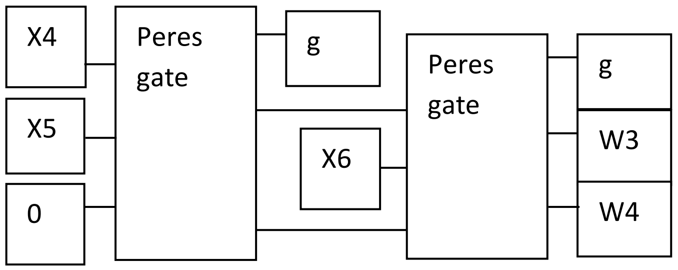
(a) Full adder F3

Fig.7. Counter 6:3 circuit using Reversible logic gates

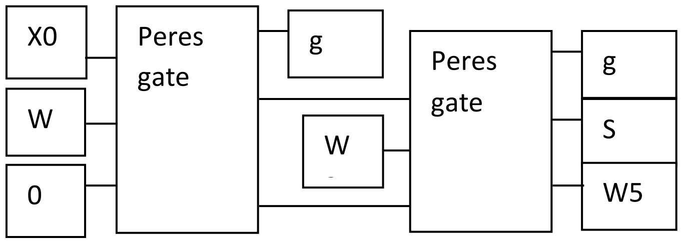
The circuit depicted in figure 6(a) is the 6:3 counters is implemented using reversible logic gates, Peres gates to be precise. The figure 7 shows the half and full adders using peres gates and the intermediate connections. X1 to X5 are the inputs to the circuits. The garbage outputs are depicted using "g". W1 to W5 are the intermediate wires. S is the sum, C1 and C2 is the carry.



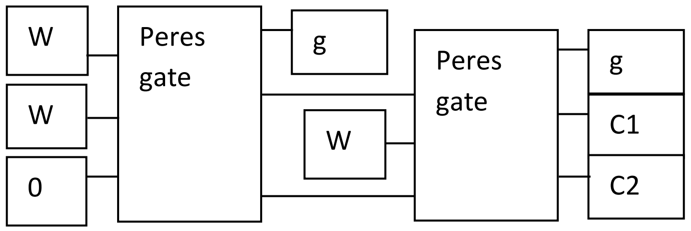
(a) Full adder F1



(b) Full adder F2



(c) Full adder F3



(d) Full adder F4

Fig.8. Counter 7:3 circuit using Reversible logic gates.

The circuit depicted in figure 6(a) is the 7:3 counter is implemented using reversible logic gates. The figure 8 shows the full adders F1 to F4 using peres gates. X1 to X5 are the inputs to the circuits. The garbage outputs are depicted using "g". W1 to W5 are the intermediate wires. S is the sum; C1 and C2 are the carry generated.

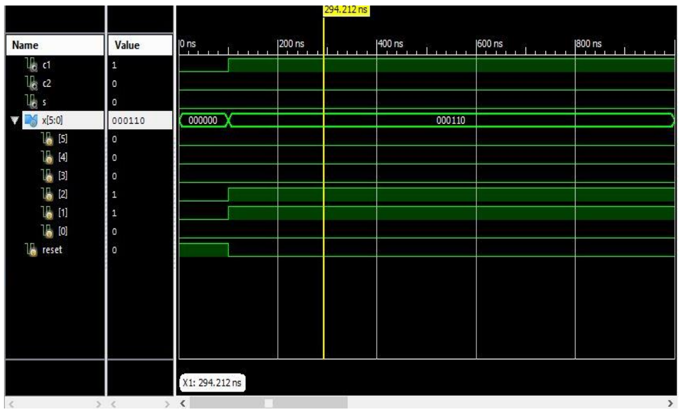


Fig. 9: Simulation results of proposed 6:3 counter using reversible logic gates.

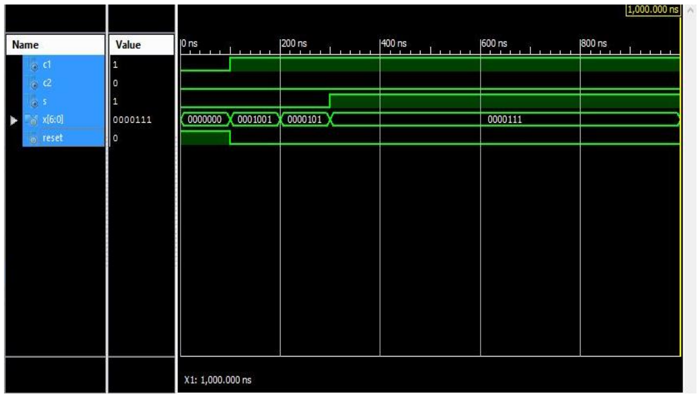


Fig.10: Simulation results of proposed 7:3 counter using reversible logic gates.

V. CONCLUSION

In this brief, a new binary counter based reversible logic is proposed. We showed that this counting method can be used to implement 6:3 and 7:3 counters, which can be used in any binary multiplier circuit to add the partial products. Heat dissipation is a major problem in the designing of a digital circuit. Rolf Landauer has proved that the information loss in a digital circuit is directly proportional to the energy dissipation. The mathematical formulation of the above statement is that if a bit is lost in a digital system, the amount of energy lost is either equal or greater than KTln2 joules in the form of heat. To provide a solution for this problem statement, reversible computing was introduced. The proposed 6:3 and 7:3 counters produced expected results and the addition of reversible logic formulation further increased the efficiency.

VI. REFERENCES

[1] C. S. Wallace, "A suggestion for a fast multiplier," IEEE Trans. Electron. Comput., vol. EC-13, no. 1, pp. 14–17, Feb. 1964.

[2] L. Dadda, "Some schemes for parallel multipliers," Alta Freq., vol. 34, pp. 349–356, May 1965.

[3] Z. Wang, G. A. Jullien, and W. C. Miller, "A new design technique for column compression multipliers," IEEE Trans. Comput., vol. 44, no. 8, pp. 962–970, Aug. 1995.

[4] M. Mehta, V. Parmar, and E. Swartzlander, "High-speed multiplier design using multi-input counter and compressor circuits," in Proc. 10thIEEE Symp. Comput. Arithmetic, Jun. 1991, pp. 43–50.

[5] S. Asif and Y. Kong, "Design of an algorithmic wallace multiplier using high speed counters," in Proc. IEEE Comput. Eng. Syst. (ICCES),Dec. 2015, pp. 133–138.

[6] S. Veeramachaneni, L. Avinash, M. Krishna, and M. B. Srinivas, "Novel architectures for efficient (m, n) parallel counters," in Proc. 17th ACM Great Lakes Symp. VLSI, 2007, pp. 188–191.

[7] S. Veeramachaneni, K. M. Krishna, L. Avinash, S. R. Puppala, and M. B. Srinivas, "Novel architectures for high-speed and low-power 3-2, 4-2 and 5-2 compressors," in Proc. 20th Int. Conf. VLSI Design Held Jointly 6th Int. Conf. Embedded Syst. (VLSID), Jan. 2007, pp. 324–329.

[8] V. G. Oklobdzija, D. Villeger, and S. S. Liu, "A method for speed optimized partial product reduction and generation of fast parallel multipliers using an algorithmic approach," IEEE Trans. Comput., vol. 45, no. 3, pp. 294–306, Mar. 1996.